Jaqpot Quattro: A Novel Computational Web Platform for Modeling and Analysis in Nanoinformatics

Chomenidis, C., Drakakis, G., Tsiliki, G., Anagnostopoulou, E., Valsamis, A., Doganis, P., Sopasakis, P., & Sarimveis, H. (2017). Jaqpot Quattro: A Novel Computational Web Platform for Modeling and Analysis in Nanoinformatics. *Journal of Chemical Information and Modeling*, *57*(9), 2161–2172. https://doi.org/10.1021/acs.jcim.7b00223

Published in:
Journal of Chemical Information and Modeling

Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
Link to publication record in Queen's University Belfast Research Portal

Jaqpot Quattro: A Novel Computational Web Platform for Modeling and Analysis in Nanoinformatics

Charalampos Chomenidis[1], Georgios Drakakis[1], Georgia Tsiliki[1], Evangelia Anagnostopoulou[1], Angelos Valsamis[1], Philip Doganis[1], Pantelis Sopasakis[2] and Haralambos Sarimveis[1,]*

1. School of Chemical Engineering, National Technical University of Athens, Heroon Polytechneiou 9, Zografou, Athens, Greece.
2. KU Leuven, Department of Electrical Engineering (ESAT), STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics, Kasteelpark Arenberg 10, 3001 Leuven, Belgium.

* corresponding author email: hsarimv@central.ntua.gr

ABSTRACT:
Engineered nanomaterials (ENMs) are increasingly infiltrating our lives due to their applications across multiple fields. However, ENM formulations may result in the modulation of pathways and mechanisms of toxic action that endanger human health and the environment. Alternative testing methods such as *in silico* approaches are becoming increasingly popular for assessing the safety of ENMs as they are cost- and time- effective. Additionally, computational approaches support the industrial safer-by-design challenge and the REACH legislation objective of reducing animal testing. Due to the novelty of the field, there is also an evident need for harmonisation in terms of databases, ontology and modeling infrastructures. To this end, we present Jaqpot Quattro, a comprehensive open source web application for ENM modeling with emphasis on predicting adverse effects of ENMs. We describe the system architecture and outline the functionalities which include nanoQSAR modeling, validation services, read across predictions, optimal experimental design and inter-laboratory testing.

Nanotechnology in recent years has attracted numerous researchers, primarily due to its impact in multiple fields from microelectronics all the way to therapeutics. However, it quickly became evident that engineered nanomaterials (ENMs) may result in unforeseen secondary effects *via* the modulation of unexpected pathways and toxic mechanisms. This has led to efforts being focused on risk management and safety assessment of ENMs, as they are now produced in increased volumes and influence our daily lives in the form of food,[1] medicine,[2] cosmetics,[3] agricultural products[4] etc. Alternative testing methods, such as *in silico* approaches, can play a central role in assessing risk of ENMs as they require less time, are cost-effective compared to their alternatives, and can be used by risk assessors/regulators, but also by the industry for self-regulation and for designing new safer products of reduced risk.[4-8]

The establishment of *in silico* approaches in the field requires standardisation and harmonisation of data, ontologies and modeling infrastructures, since the collection, sharing, and processing of data in order to extract useful information and develop predictive models are currently not consistent.[9] JaqPot Quattro (JQ) is the first comprehensive computational platform specifically developed for ENM *in silico* modeling in the context of the eNanoMapper FP7 funded project (http://enanomapper.net/). JQ is offered to the community as an open source platform, which is not limited to developing and sharing predictive models for adverse effects, but also contains functionalities that support and evaluate experimental work and collaboration among modeling and computational research groups.[10] Its open REpresentation State Transfer (REST) based architecture and Application Programming Interfaces (APIs) allow integration with third-party services and the continuous development and extension of the platform by both the original developers and the entire community.

The system has direct access to all data contained in the eNanoMapper data repository, including raw data such as images, spectral data, omics data etc., which allows easy integration of newer data as they become available and full characterization of complex nanoparticle structures.[11] The user-friendly interface allows non ICT-experienced users (such as toxicologists, regulators etc.) to have easy access to all modeling JQ tools and functionalities, but also to a repository of validated models that have been published in the literature or have been accepted for regulatory purposes. Using an OpenSSO/OpenAM authentication and authorization service, different access levels can be assigned to different users. Therefore, the JQ user-interface provides the community with a sustainable ready-to-use framework and application that can be used for disseminating predictive models and tools. This work briefly presents the JQ architecture and focuses on the end-user point of view through a more detailed description of the user-friendly interface.

Results and Discussion

JQ is a publicly available open source web application for modeling and analysis of ENMs. It is built on several programming languages, licensed with the GNU GPL v3 license and consists of a multitude of independent web services that follow the modern and scalable microservice architectural style. A schematic overview of the JQ architecture is presented in Figure 1. At the core of the project lies a central API Gateway implemented in Java 8 and JEE7 that manages the flow of requests, wraps long running procedures to asynchronous tasks, provides authentication and authorisation mechanisms, and acts as a central RESTful Web endpoint for the whole system. Documentation for this public API is presented as a simple web page at http://app.jaqpot.org:8080/jaqpot/swagger/, which demonstrates all RESTful resources contained by JQ, the HTTP Verbs that can be applied on each resource, as well as all supported

representations, as shown in Figure 2. More information about the system architecture and integration with third-party services are given in the *Methods* section.

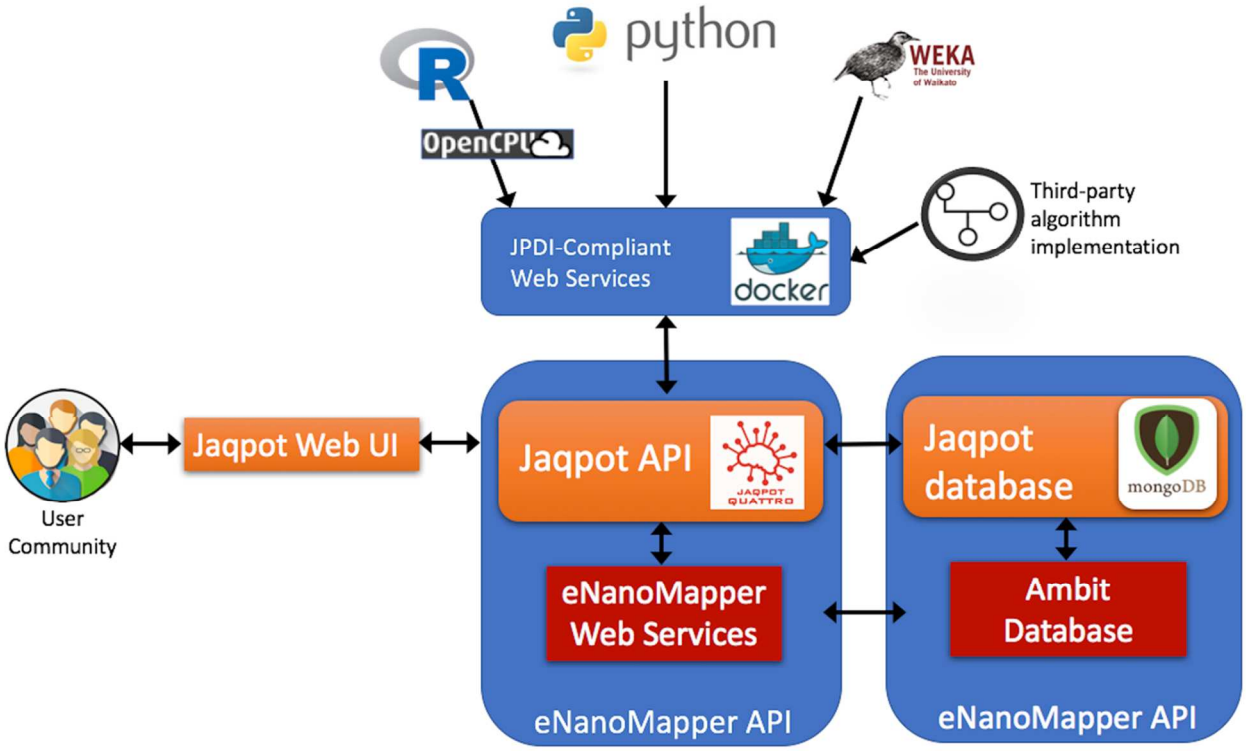

Figure 1. Schematic overview of JQ architecture.

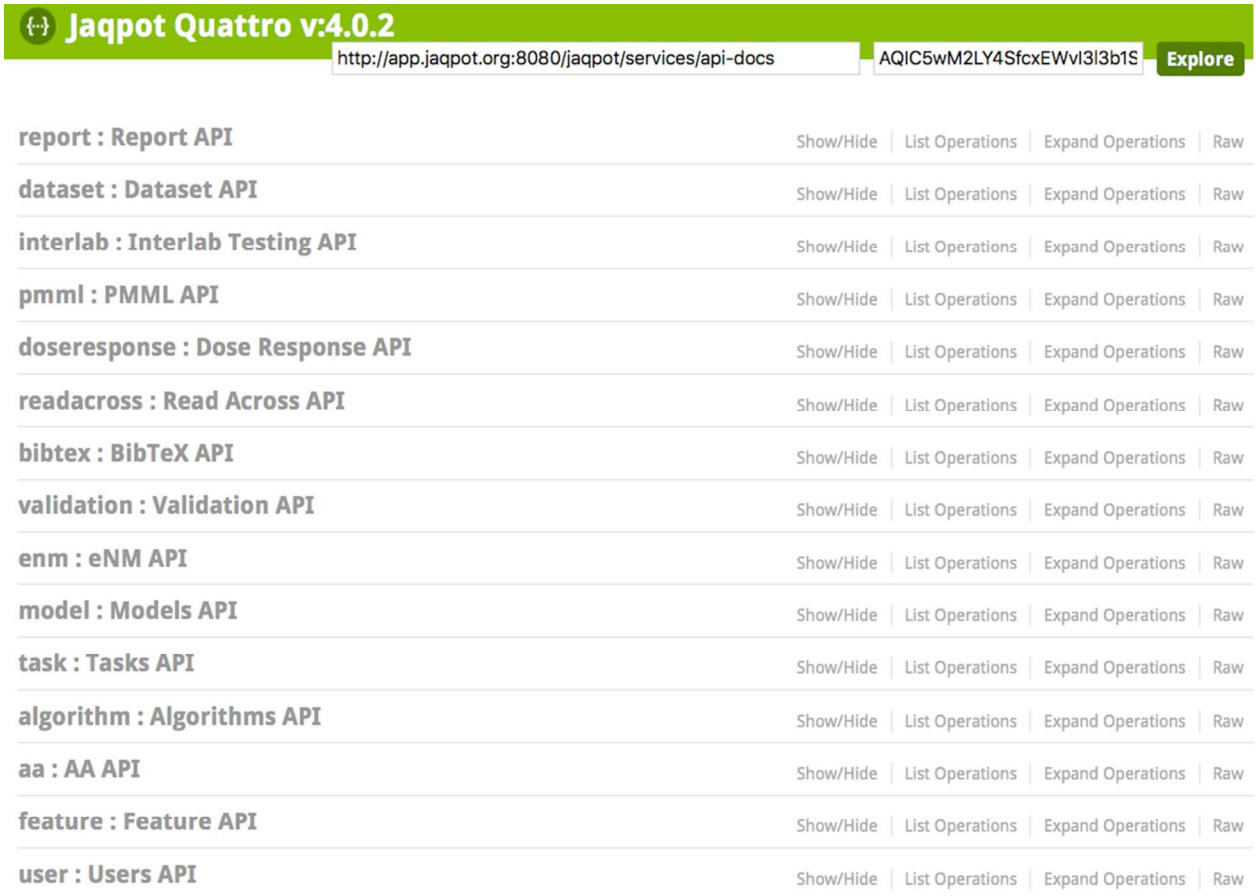

Figure 2. Documentation of JQ API using the Swagger tool.

The user interface of JQ can be accessed at http://www.jaqpot.org/ (Figure 3) and includes the following functionalities, which will be elaborated in the rest of this section:

- Allows users to import, select and process data from the eNanoMapper database service and transform them to tabular forms that can be used for modeling purposes.
- Offers various options of data preprocessing, such as standardisation, normalisation, variable selection and usage of Predictive Model Markup Language (PMML) documents to allow further data transformations.
- Integrates descriptor calculation services based on raw data (images, omics data) or crystallographic data (MOPAC descriptors)
- Supports both nano quantitative structure–activity relationship (nanoQSAR) and read across predictive approaches by integrating major statistical and data mining algorithms from third-party open-source services, but also allowing the users to create their own custom algorithms and methods.
- Includes rigorous model validation schemes: split-, cross- and external validation services.
- Offers a repository of well validated predictive models published in the literature as ready to use publically accessible web applications, where the users can upload their own data and obtain the model predictions.

- Provides free private space to the users for storing and securing their resources (datasets, models, reports). Users can safely access their private resources from any computer or electronic device around the world equipped with Internet connection and a web browser.
- Supports collaborative and integrated work among experimental and modeling groups by offering factorial and iterative optimal experimental design, as well as interlaboratory proficiency testing services.
- Incorporates model validation and interlaboratory reporting services as well as automatic generation of QSAR Prediction Reporting Format (QPRF) editable reports, for dossier submissions to IUCLID5. All reports can be downloaded as PDF files.

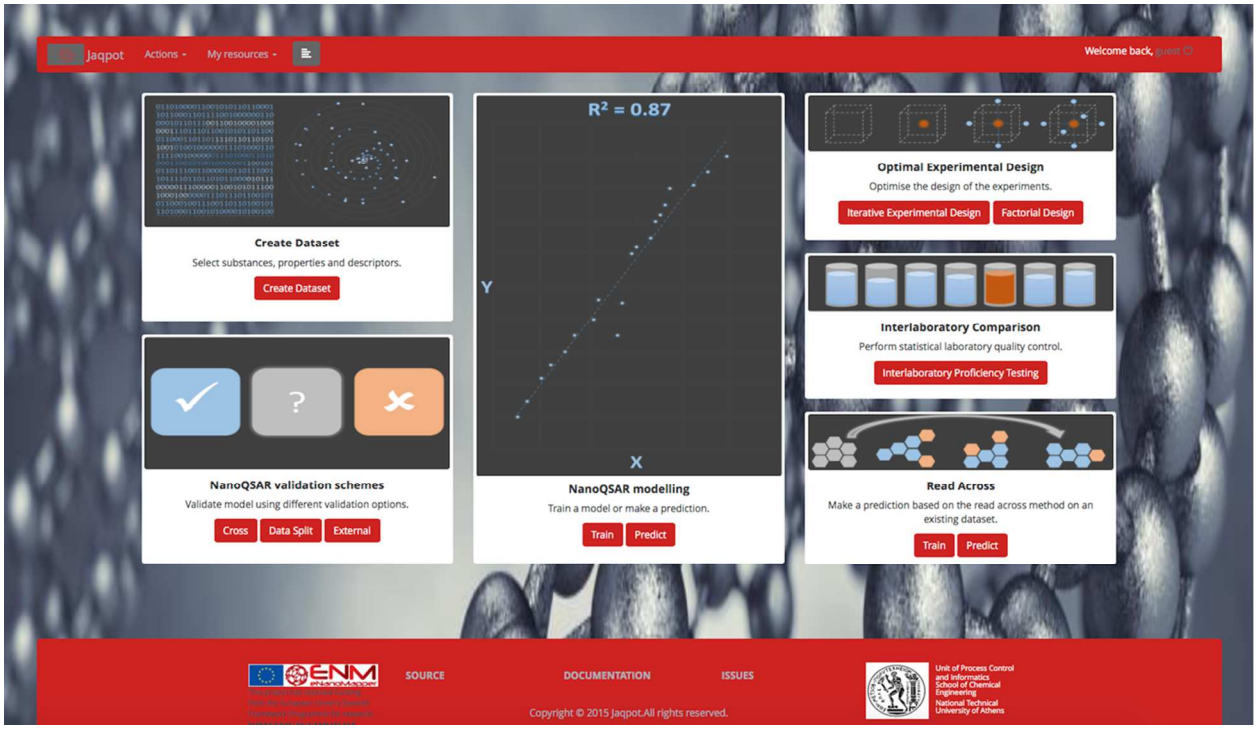

Figure 3. JQ user interface, available at www.jaqpot.org.

Accessing the JQ functionalities: After clicking on the http://www.jaqpot.org/ link for the first time, the user enters the central JQ page. For accessing the different functionalities, he/she has two options. The first option is to sign in as a guest using the string "guest" as both user ID and password. The second option is to create a new account by clicking on the "Create Account" button which opens the OpenTox registration form at http://old.opentox.org/join_form. Both options give the user access to all JQ functionalities. The only difference is that the resources (datasets, models, reports) created by the "guest" account are visible and can be manipulated by all users accessing the system with the same credentials, while by creating a personal account, the user is offered a free private space in the server, which is not visible to other users.

Creating datasets. The first available option for the user is to create a dataset on which to perform modeling or other tasks. This service allows the retrieval of data from the eNanoMapper data repository (https://apps.ideaconsult.net/enanomapper/) by selecting a dataset creator. Users may select all or a subset of ENMs along with the experimental properties and calculated

descriptors that will be included in the dataset. Experimental properties are grouped into four categories, namely physicochemical, environmental fate, human toxicity and environmental toxicity properties. The user can select entire groups or particular experimental properties in each group. As far as calculated descriptors are concerned, JQ is currently integrated with two descriptor calculation services:

- The MOPAC[12] service, which computes the following semi-empirical quantum mechanical descriptors, should a PDB file be available in the eNanoMapper data repository that accompanies the characterisation of an ENM: Electronic energy, Core-Core repulsion, Molecular weight, EHOMO, ELUMO, No. of filled levels, Total energy, Final Heat of Formation, Ionization Potential. Calculations are based on the MOPAC service available at https://data.enanomapper.net/algorithm/ambit2.mopac.MopacOriginalStructure and can also be accessed with REST commands.
- The image analysis service developed in-house as a web application that computes descriptors based on image processing, should an electronic image file be available. This service has integrated functionalities and descriptors from the ImageJ[13] library (such as Bounding Rectangle, Centre of Mass, Centroid, Feret Diameter, Fit Ellipse, Integrated Density, Kurtosis, Mean Grey Value, Median, Min and Max Grey Levels, Modal Grey Value, Perimeter, Shape Descriptors), but includes additional image descriptors for ENMs proposed by Gajewicz *et al.*[14] (Anisotropy Ratio, Area, Circularity, Equivalent volume diameter, Equivalent volume to surface, Porosity, Sphericity, Surface diameter and Volume). The service can be accessed independently from the JQ platform at http://app.jaqpot.org:8880/imageAnalysis/.

Training a nanoQSAR model. Users can train a nanoQSAR model using either a dataset created by them or one of the example datasets provided by JQ. The training process starts with the selection of the dataset, proceeds to the algorithm selection and finally allows parameterization, which ranges from setting the available algorithm parameters all the way to variable selection, scaling of the dataset values, using Predictive Model Markup Language (PMML) transformations and setting an algorithm for defining the domain of applicability. Among the variables contained in the data set, the user selects the end-point variable and the input variables. Uploading a PMML file allows the user to define a "data dictionary" and a "transformations dictionary": The latter defines mathematical formulas to be applied on the variables selected by the former. The predictive model will then be trained using the transformed variables as input. For example, the user can apply a nonlinear transformation on a single variable (such as square root, logarithm, exponential function) or multiple variables (taking for example their product or summation). The user is prompted to set a name and description of the model. Once the nanoQSAR model is created, it is saved in the user's private space, and it can be immediately applied for providing predictions or it can be accessed at any later time through a web browser. For most algorithms, a PMML representation of the model is created that allows exchange, transfer and sharing of the model with other computing platforms and PMML compliant applications. The current JQ release has been integrated with the R and Python languages as well as the WEKA data mining platform and provides implementations of most major statistical and machine learning regression and classification algorithms: linear, multiple linear and Lasso regression; Partial Least Squares (PLS) with and without Variable Importance in Projection (VIP) scores; Support Vector Machines (SVM); Generalised, Multinomial and

Bernoulli Naive Bayes and Iterative Dichotomiser 3 (ID3) and Conditional Mutual Information (CMI) decision trees.[15] Additional algorithms can be added by the user as long as they conform with JQ requirements.

Accessing a repository of validated nanoQSAR models. JQ users are provided access to well-validated nanoQSAR models that have been presented in the literature. Each model is offered as a ready-to-use web application and contains all the important model information: Title of publication, DOI, extended description, contributors, publishers and keywords. For demonstration purposes, two established and popular nanoQSAR models are currently included in the model repository. Firstly, a linear regression model that predicts toxicity ($\log(1/LC_{50})$) of Metal Oxides (MeOx) in the human keratinocyte cell line (HaCaT), using variables selected by Gajewicz *et al*.[14] The model has been developed using a training dataset of 10 MeOx ENMs[16]. Secondly, a PLS regression model trained on a protein corona dataset consisting of 76 proteins for 84 gold ENMs of different diameters (15, 30 or 60nm) and different coatings that predicts cell association of gold ENMs with human A549 cells.[17] We plan to continuously update the model repository with all major developments in the field, so that researchers in the area will use JQ as a central point for searching for relevant nanoQSAR models and applying them on their own data.

Use nanoQSAR model for obtaining predictions. JQ models are linked-data entities. Users can select a customised model or an existing one from the repository and use it on their own data to generate predictions. The result of the prediction service is an automatically created web page that contains the model predictions and the index corresponding to the domain of applicability, if such an algorithm has been selected by the users during the training phase. By clicking on the QPRF button, a complete QPRF report is generated. QPRF is a harmonised reporting template for summarising and reporting substance-specific predictions generated by QSAR/nanoQSAR models, that addresses the need of using such predictions in the regulatory assessment of chemicals (*i.e.* the REACH objective to reduce animal testing).[18]

Read-across approach for obtaining predictions. Another service provided by JQ is making read-across predictions based on the similarity of query substances to instances in the training set. Read-across is used to predict properties of substances when only few experimental data are available.[19] Similarly to nanoQSAR modelling, in order to train a read-across model, the user must provide a training dataset and specify the input variables and the end-point to be predicted. Additionally, the user must select a distance metric (currently available: Euclidean, Manhattan or ensemble), a distance threshold between zero and one, and whether he/she would like to compute a confidence level when making a prediction. The distance metric is used for identifying neighbors (similar substances) to the query substance. In order to make a prediction for a query ENM with unknown end-point value, the method first calculates the distance between the query instance to all instances contained in the training set based on the selected distance metric. Training instances with distance above the threshold are discarded. The read-across prediction is computed as a weighted function of the known end-point values of the remaining training ENMs. The lower the distance of the query instance to a training instance, the higher the weight used for the prediction. Similarly, the more neighbouring instances (those used to make the prediction) the higher the confidence level of the generated prediction. The outcome of using read-across for prediction is a dataset containing the predicted values, along with the confidence levels, if the

option is selected. QPRF reports can be generated in the same fashion with reports generated for nanoQSAR model predictions. For testing the read-across model, users must simply select a validation dataset with known end-point values whose attributes are compatible with those of the training dataset. End-point predictions are compared to actual values using the same validation metrics as in nanoQSAR modelling.

Validation Services. Model validation schemes have been added to the JQ functionalities in order to provide users with the opportunity to evaluate the accuracy and robustness of their models. Three types of validation have been made available, namely training set split validation, cross validation and external validation. Briefly, training set split validation refers to the splitting of a dataset into training and test sets based on a user-defined ratio, cross-validation comprises multiple such data splits *i.e.* 5-fold cross validation means that the data will be split 5 times to create five 80%-20% training-to-test set splits, and external validation requires holding out a dataset from the modeling process in order to use it only for testing after the model is built. In this service, all values, statistical metrics and plots for the validation services are made available as editable and downloadable reports, which are described later in this article.

Interlaboratory testing. Interlaboratory proficiency testing comprises a series of statistical tests used to assess bias in laboratories and increase repeatability of experiments according to international standards. This can be carried out by processing the reported measurement results of laboratories on either reference materials (for which an expected range of values exists), or on the consensus outcomes of a group of laboratories. All dataset entries are subjected to algorithms for defining the robust average and robust standard deviation. Then, the assigned value and uncertainty are calculated, the latter depending on whether laboratories have provided their own uncertainty estimates or not. Once these values have been calculated the process continues by calculating performance statistics and plots that allow easy visualisation of the results. Labs may receive warning or action signals, which include checking the measurement procedures, competence of staff and performing calibration of the laboratory equipment. All calculated values (robust average, standard deviation and uncertainty), plots and matrices computed (anonymised lab dictionary, detailed statistics, action/warning signals, etc.) are made available in a format, discussed in detail later in this work.

Experimental Design. Experimental design offers computational features aiming to optimally guide regulators' experiments, towards a safer by design approach. The experimental design tools allow a systematic iterative procedure between experimental and modeling groups, where predictive models suggest conditions for next series of experiments and in turn, new experimental results are used to improve the models. Full factorial designs, but also several optimal experimental design options, such as D-optimal, A-optimal and I-optimal designs can be used and are fully integrated in the iterative procedure. The iterative experimental design scheme implemented is based on the work published by Brandmaier *et al*.[20-21], an adaptive approach that combines optimal criteria with PLS and leave-one-out cross validation or principal component analysis (PCA).

Reporting Services. As briefly discussed above, JQ produces a variety of reports automatically generated within its infrastructure, such as QPRF reports for any prediction, as well as detailed statistical and graphical outcomes for all validation schemes and interlaboratory testing. These

consist of three types of entities: single values, array values and plots. Single entities include general information such as title, author, report type, date created, as well as single calculated values such as R^2 and standard error in validation reports, or robust average and uncertainty in interlaboratory testing reports. Array values comprise matrices, such as full prediction tables (validation), detailed statistics (interlaboratory testing), or substance information (QPRF reports). Plots are PNG images containing vital visualisations in order to make the report calculations easily interpretable by the users, which include figures such as QQ plots, bias histograms, 3D PCA plots, Confusion matrix etc. depending on the type of report. Every entry except plots can be manually edited and saved online. This feature is highly useful for retrospectively adding descriptions post-modeling or manually adding data that is not available in the eNanoMapper database but can be found from additional sources (*i.e.* the EC number of a nanoparticle to be added into a QPRF report). Finally, all reports are downloadable as PDFs, in order to allow convenient offline storage to the users.

Training Material. A series of tutorials, training videos and manuals are now available stepping from importing the data, model creation, validation, evaluation and visualisation (http://www.jaqpot.org/documentation, http://www.enanomapper.net/library). Also available are example workflows that cover some of JQ's key features (workshops material). Additionally, important training material is included in the project's Github repository https://github.com/KinkyDesign/Jaqpot-Training-Material.

Case study: Using Jaqpot Quattro for nanomaterial modeling

Exploration of nanomaterial models using JQ can be carried out with great flexibility at each stage, through an easy-to-navigate tool. In this case study, we demonstrate the replication of computational results from Gajewicz and coworkers,[14] from the selection of input substances and ENM properties, to the construction of a linear nanoQSAR model and the creation of validation and QPRF reports. As mentioned above, this model is also included in the model repository of JQ. The complete dataset contains information for 18 MeOx ENMs and the end-point to be predicted is $\log(1/LC_{50})$ toxicity to the human keratinocyte cell line (HaCaT) cell line, which is a common in vitro model for keratinocyte response during toxic dermal exposure. The dataset was split into training and validation sets (consisting of 10 and 8 instances respectively) as outlined by Gajewicz *et al.*[14] We used exactly the same dataset to demonstrate the JQ read-across functionalities, by building a read-across model. The models were created using the "guest" account, so they are visible and accessible to all users entering JQ as guests.

Building a nanoQSAR model. By selecting the "Train" option in the "NanoQSAR modelling" box on the main menu (or "Train a Model" under Actions), we selected the dataset containing the training data, which can be found under "Example Datasets" with the description "10 MeOx NPs with 29 descriptors, used for predicting HaCaT toxicity". Subsequently, we chose the "Linear Regression" algorithm (Figure 4) and the input model attributes:

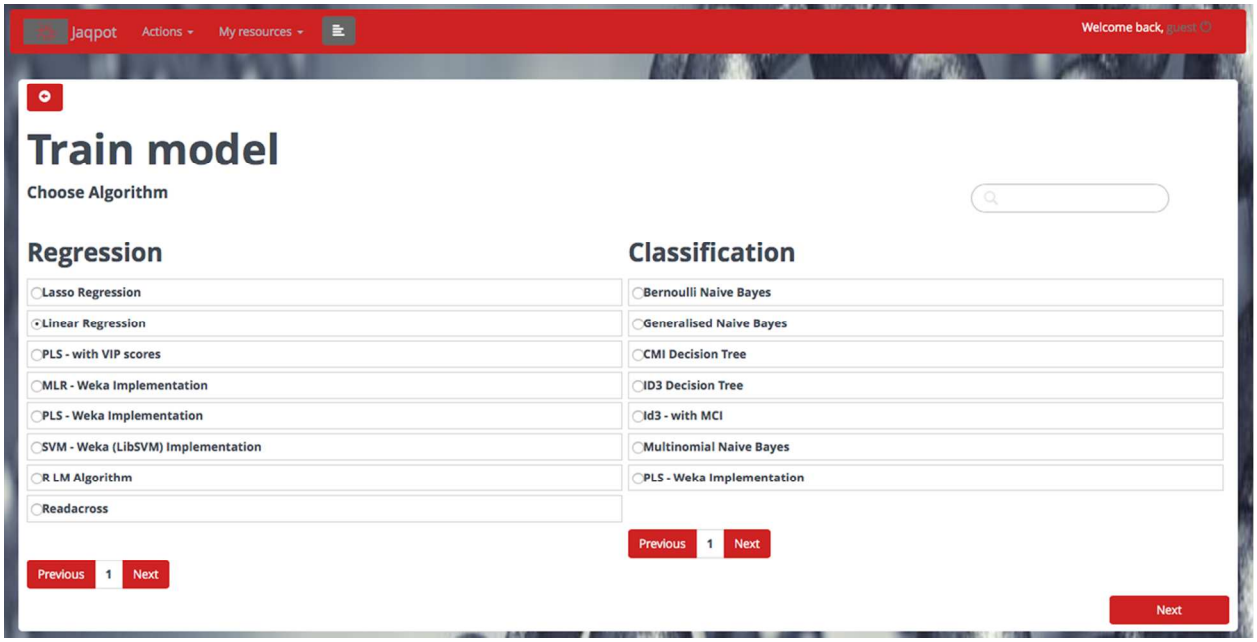

Figure 4. Selection of training algorithm.

- Model title: Linear Model to Predict log(1/LC50) to HaCat for metal oxides
- Model description: The linear model predicts log(1/LC50) to HaCaT for metal oxides using Mullikens electronegativity and Standard enthalpy of formation as input parameters
- Input variables: “Mulliken Electronegativity” and “Standard Enthalpy of formation of metal oxide nanocluster” selected
- Output variable: “log(1/LC50)” selected
- Scaling preferences: “Normalisation” selected
- Domain of Applicability calculation: “Leverage method” selected

as shown in Figure 5.

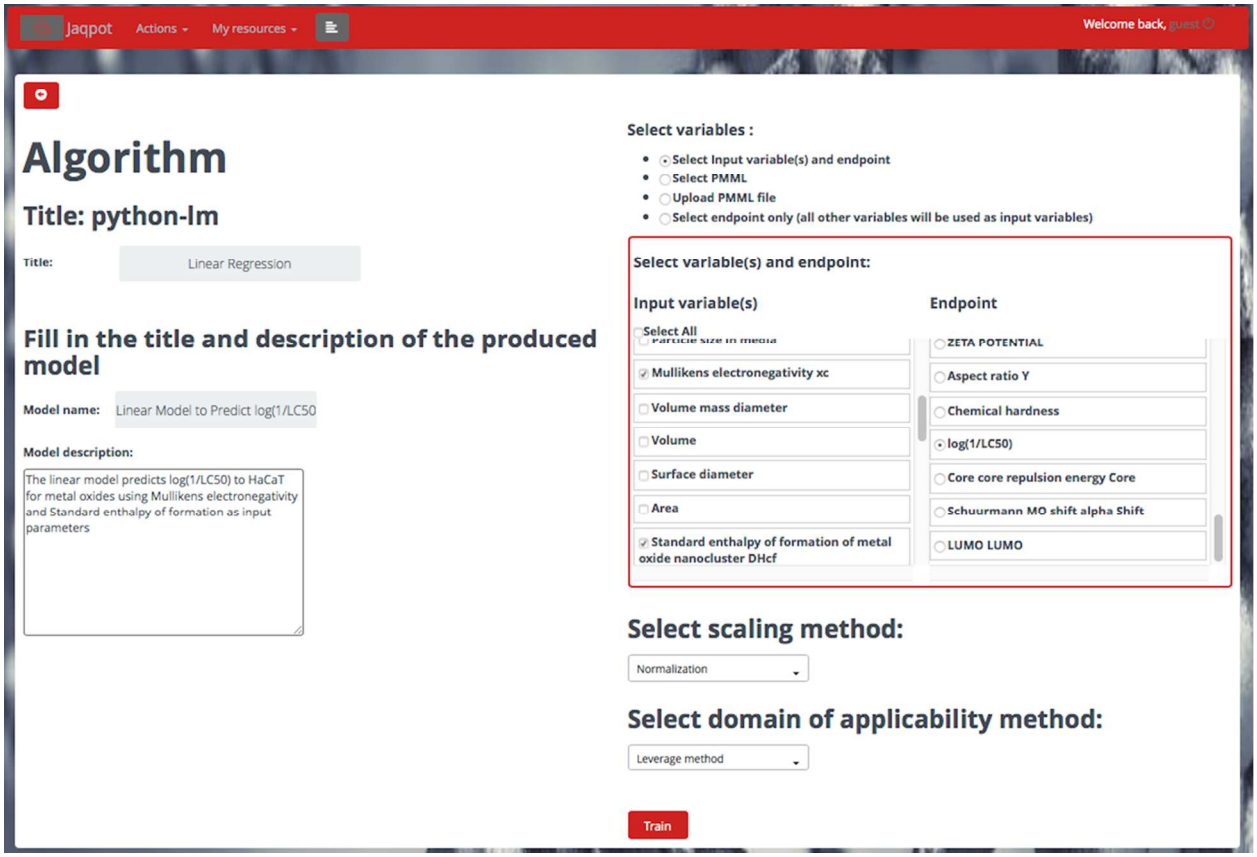

Figure 5. User interface for definition of model title, description, input/output variables, scaling preferences and domain of applicability calculations.

Once completed, a model was produced (Figure 6) that constitutes a ready-to-use web resource: http://www.jaqpot.org/m_detail?name=4Paf6qHXYAidsM0q4Ahc. At this page, apart from examining the model properties (input/output data, algorithm used etc.) users may validate the model or make predictions.

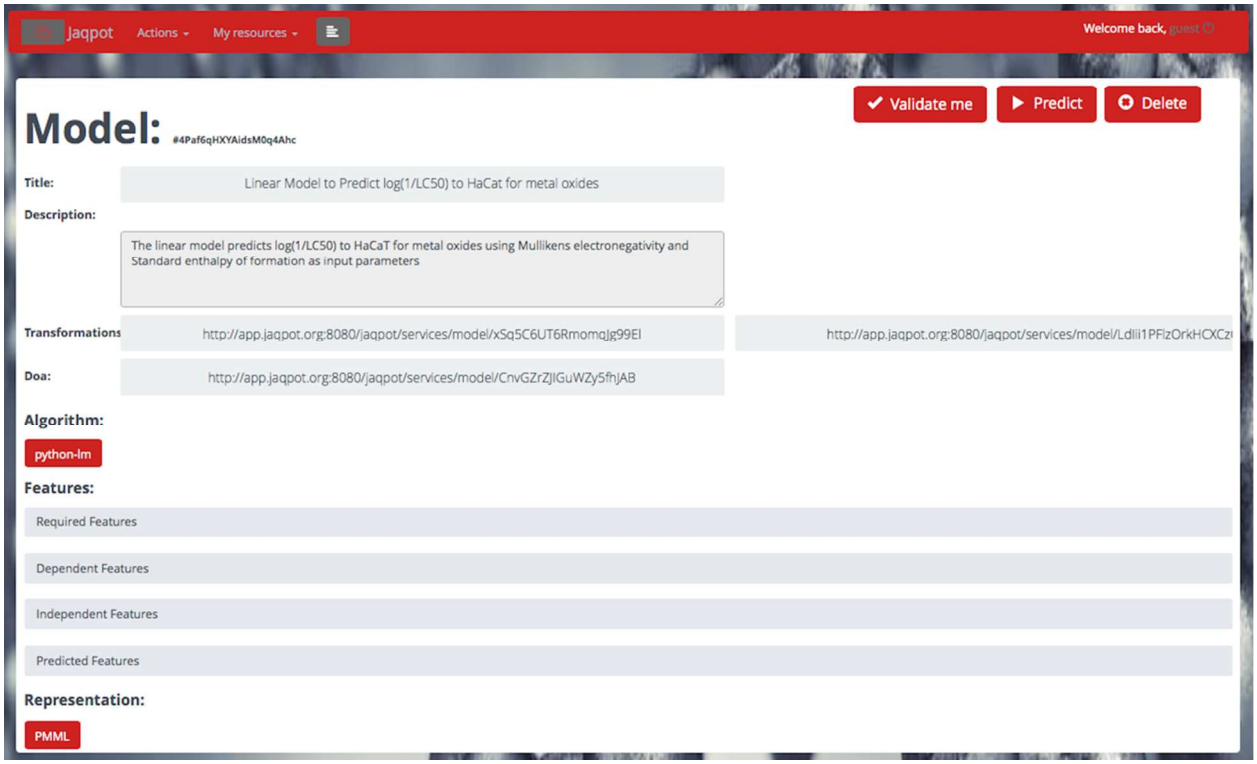

Figure 6. Model page containing ID, Title, Description, Required/ Dependent/ Independent /Predicted features, PMML download (when available) and Validation, Prediction and Deletion options.

nanoQSAR model validation . In order to validate the model, we applied external validation on the test dataset from Gajewicz *et al.*[14] which has been made available under "Example Datasets" with the description "8 MeOx NPs with 29 descriptors, used for predicting HaCaT toxicity". This produced a detailed Validation Report (URI: http://www.jaqpot.org/report?name=QVWHQgjbpZqywOB) providing performance metrics, a full table of real and predicted values and plots. A screenshot of this report is shown in Figure 7.

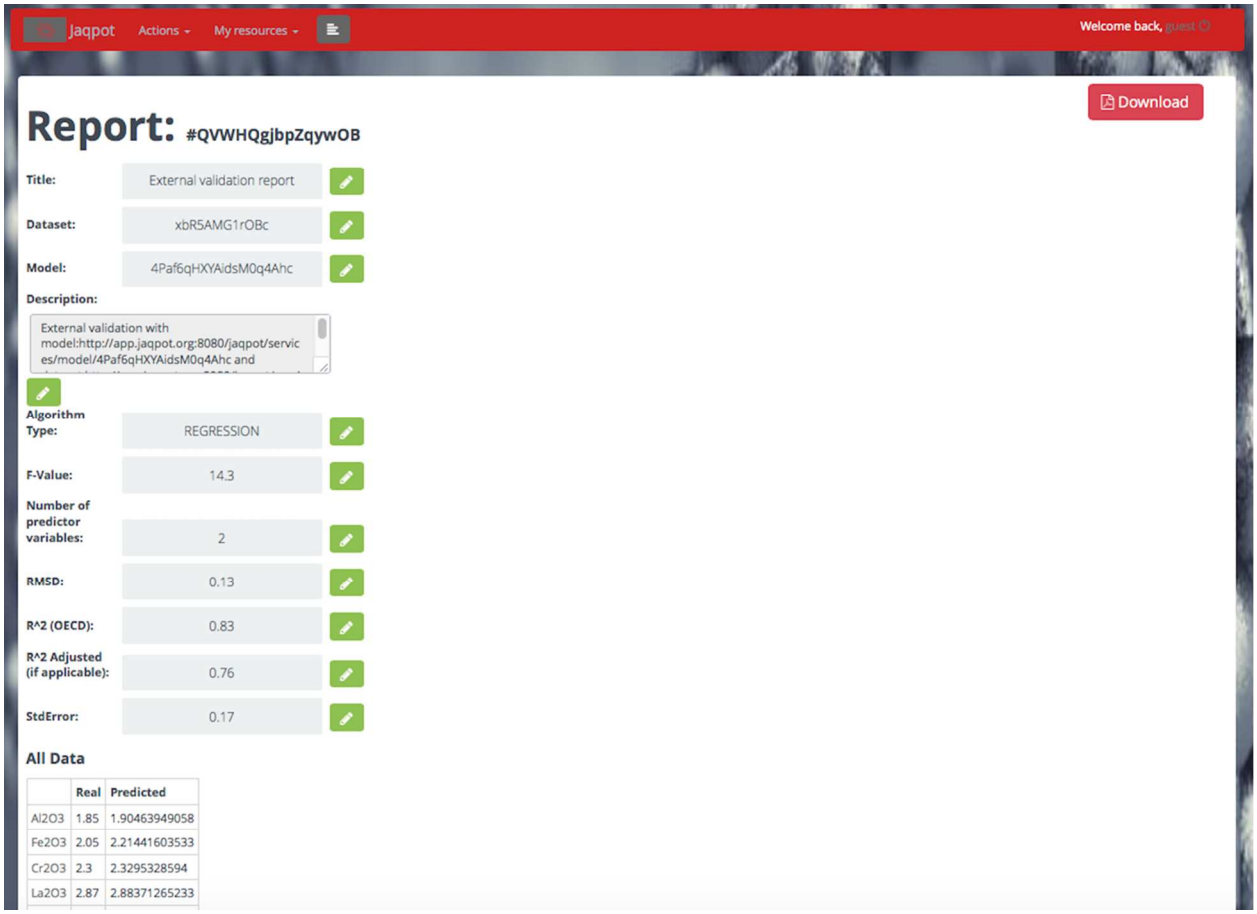

Figure 7. A screenshot of the model validation report. Contains performance metrics, full real versus predicted values for endpoint table and plots.

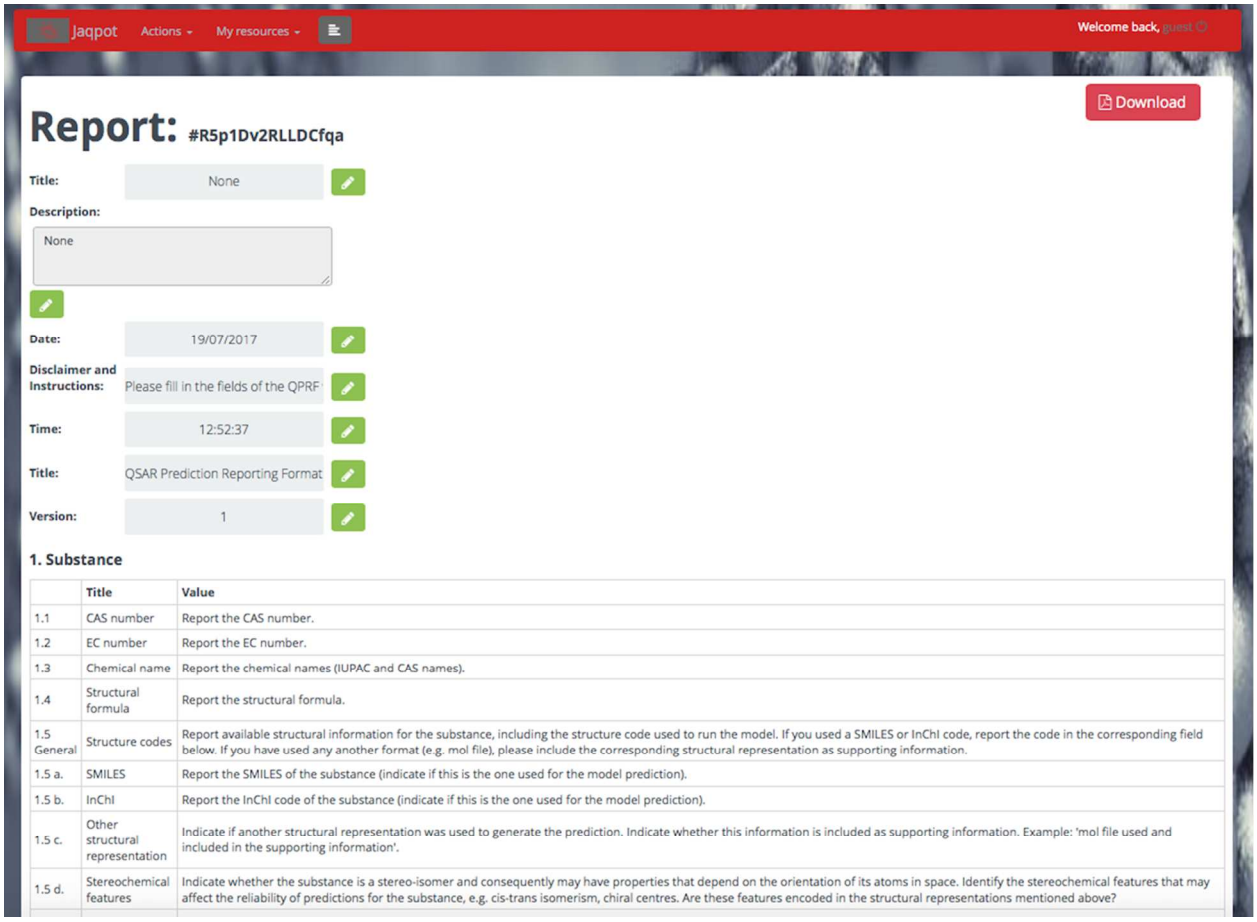

Figure 8. A screenshot of the QPRF report. Contains information fields substance, general, prediction and adequacy, which are filled in automatically when data is available or given default values (instructions) when not.

Use of nanoQSAR model for making a prediction. From the model webpage we selected the "Predict" option and then the "insert values" option, to use the model for predicting the end-point value for an unknown metal oxide with Mulliken Electronegativity and Standard Enthalpy of formation equal to 5eV and -3000Kcal/mol respectively. The following link was produced: http://www.jaqpot.org/predicted_dataset?name=2rhsusjBMeHwc8T1xAwT&model=KchmtK9UwIYlByRR6Kux showing a predicted value of 2.45 for the end-point along with a positive leverage domain if applicability (DoA) value, which means that the new substance is within the domain of applicability of the model and thus, the prediction can be considered reliable. By scrolling the cursor to the right and clicking on the "QPRF Report" button, a QPRF report was generated for the new substance and is made available at http://www.jaqpot.org/qrf_report?uri=compound1&dataset=2rhsusjBMeHwc8T1xAwT A screenshot of this report is shown in Figure 8.

Training and validating a read-across model and using it for making predictions. The workflow is very similar to the one used for nanoQSAR modelling and will be presented briefly using exactly the same training and validation set used in the nanoQSAR case study. The workflow starts by selecting the "Train" option in the "Read Across" box on the main menu (or "Read Across Train" under Actions). A read-across model was generated by selecting the

Manhattan distance and a threshold value equal to 0.15 and choosing the "Nearest Neighbor Confidence" method for calculating the confidence value. The model title was "Read-across Model to Predict log(1/LC50) to HaCat for metal oxides" and the model description was "The read-across model predicts log(1/LC50) to HaCaT for metal oxides using Mullikens electronegativity and Standard enthalpy of formation as input parameters". The produced model is available at http://www.jaqpot.org/m_detail?name=xGeGbcs2o5nQ8U3u4A3B. By validating the model on the 8 MeOx validation set, a validation report was generated and can be reached at http://www.jaqpot.org/report?name=1VjFnQteVr5oP5h. Finally the model was applied on the unknown compound with Mulliken Electronegativity and Standard Enthalpy of formation equal to 5eV and -3000Kcal/mol respectively and generated the prediction and the QPRF reports that are available at http://www.jaqpot.org/predicted_dataset?name=D8nyt68Y6jMmyZE4wpy0&model=xGeGbcs2o5nQ8U3u4A3B and http://www.jaqpot.org/qrf_report?uri=compound1&dataset=D8nyt68Y6jMmyZE4wpy0 respectively. The predicted end-point value was 2.37, and the confidence value was 0.91, which means that the read-across prediction can be considered reliable.

Conclusions

Effective modeling of ENM toxicity is particularly useful from both the regulators and the industrial points of view, as it can speed up the process of assessing risks of ENMs, can reduce and guide time and cost intensive wet-lab experiments and can assist in designing safer ENMs. However, ENM modeling is still an underdeveloped discipline mainly due to the lack of a standardised commonly agreed terminology, curated, user-friendly searchable databases and a computational infrastructure that supports cross-talk and interaction between diverse sources of data.[7] This paper has focused on the JQ modeling infrastructure, which along with a comprehensive ontology and a modular infrastructure for data storage, sharing and searching (all developed and cross linked in the context of the FP7 funded eNanoMapper project) are designed to fill the gaps mentioned before. JQ already supports many of the computational needs in the field of computational nanotoxicology, but its open source nature and its design architecture will allow the community to incorporate and integrate additional functionalities in the future. Central to the JQ development and its future extensions should be the aim to support collaborative and integrated work among experimental and modeling researchers in this emerging and very promising scientific field.

Methods

Jaqpot Quattro Application programming Interface development. JQ API is a modern level 3 RESTful API, according to Richardson's Maturity Model,[22] implemented in Java 8 and JEE7. The API acts as a central endpoint for the whole JQ system, providing communication between the microservices comprising the system and the outside world. Each JQ independent microservice, including the core API, is a purely stateless construct that allows for individual replication and balancing, thus giving the ability to scale horizontally on any part of the system. By implementing functionality in such a microservice and polyglottal way, JQ is a cloud-ready application that uses the benefits of each platform, having the ability to incorporate functionality by various established and open-source machine learning and data analysis toolkits. At the same time, it paves the way to the creation of a modeling ecosystem, where independent systems contribute and collaborate, while maintaining their autonomy, provided they adhere to the API.

The orchestration and deployment of JQ services is handled by the containerization tool *Docker*, which provides a safe and isolated space for each microservice to run, while having its health monitored by the Docker daemon. Docker Images for all JQ components can be found at https://github.com/KinkyDesign/jaqpot-docker-scripts.

The JQ API wraps long running procedures to asynchronous tasks. This is achieved by the Java Messaging System (JMS), a queueing mechanism that can be used to assign each task to a worker thread taken from a finite customizable thread pool, while also providing a simple retry on fail logic. Each task can be monitored by performing a GET request on its corresponding URI, while long polling is also supported. JQ performs communication between its services in an asynchronous way by using Apache HttpAsyncClient and Java I/O Streams. This Java Non-blocking I/O (NIO) based client allows maintaining a large number of active connections while keeping a minimal memory footprint, which gets even better when combined with on-the-fly JSON marshalling/unmarshalling by the use of Streams.

Jaqpot Quattro User Interface development. The JQ user interface is a publicly available open-source web application written in Python. The framework used was Django 1.7,[23] which follows the Model-View-Controller (MVC) architectural pattern. The URL dispatcher maps the requested URL to a view function and calls it. The view function performs the requested action, which typically involves communicating with the JQ API. The model defines the data in Python and interacts with it. A template returns HTML pages. For the design and implementation of the user interface, we used HyperTextMarkup Language (HTML) for representing information and Cascading Style Sheets (CSS) for designing the pages. In addition, JavaScript[24] and jQuery,[25] a JavaScript library designed to simplify the HTML scripting, were used for event-handling and loading new page content. These technologies were also used for validating input values of forms and submitting data to the server *via* asynchronous JavaScript and XML (AJAX),[26] without reloading the page. Because JavaScript code can be run locally in a browser, it can respond to user actions quickly, making an application more responsive. With Ajax, web applications can send and retrieve data asynchronously without interfering with the display and behaviour of the existing page. The application sends HTTP requests to the JQ API in order to perform desired actions and receives responses in JSON format. JSON is a format for saving and transporting data and is used when data is sent from a server to a web page. Users should create an account using the OpenTox platform at http://old.opentox.org/join_form or use the "guest" account in order to sign into the JQ web application.

Creating datasets suitable for modeling. JQ algorithm services require input data in a standardised format in order to generate a predictive model. Therefore, raw experimental data cannot be used directly for modeling purposes. Experimental data are, more often than not, heterogeneous by nature and properly structuring these is not a trivial task. To this end, a web service acting as a link between experimental data and data for modeling was introduced, which will be hereafter referred to as the *Conjoiner* service.[11] This service performs the task of transforming the experimental data into a modeling-friendly format, and producing standardised datasets as specified in the OpenTox API. One can initiate a Conjoiner service operation by specifying a bundle URI. A bundle is an eNanoMapper resource that acts as an assortment of experimental effects, images and molecular structures, for nanomaterials, and the Conjoiner service's job is to combine all that disparate data into a dataset suitable to be fed to an algorithm

service. Regarding experimental data, multiple individual measurements, interval-valued measurements (lower and upper values), or values accompanied by a standard measurement error, may be included for the same endpoint in a bundle. These need to be aggregated into a single value, which is currently carried out by taking the average value of all experimental measurements after having excluded outliers identified by a Dixon's q-test. Other aggregation procedures based on more elaborate outlier detection criteria and rejection/aggregation schemata can also be implemented. When bundles contain links to raw data of nanomaterials such as images, crystallographic data and proteomics data, descriptor calculation web services are activated to produce the calculated descriptors. Data as heterogeneous as chemical structures, raw experimental measurements, spectra and microscopy images can be combined by the Conjoiner service (Figure 9) to produce a dataset for modeling purposes.

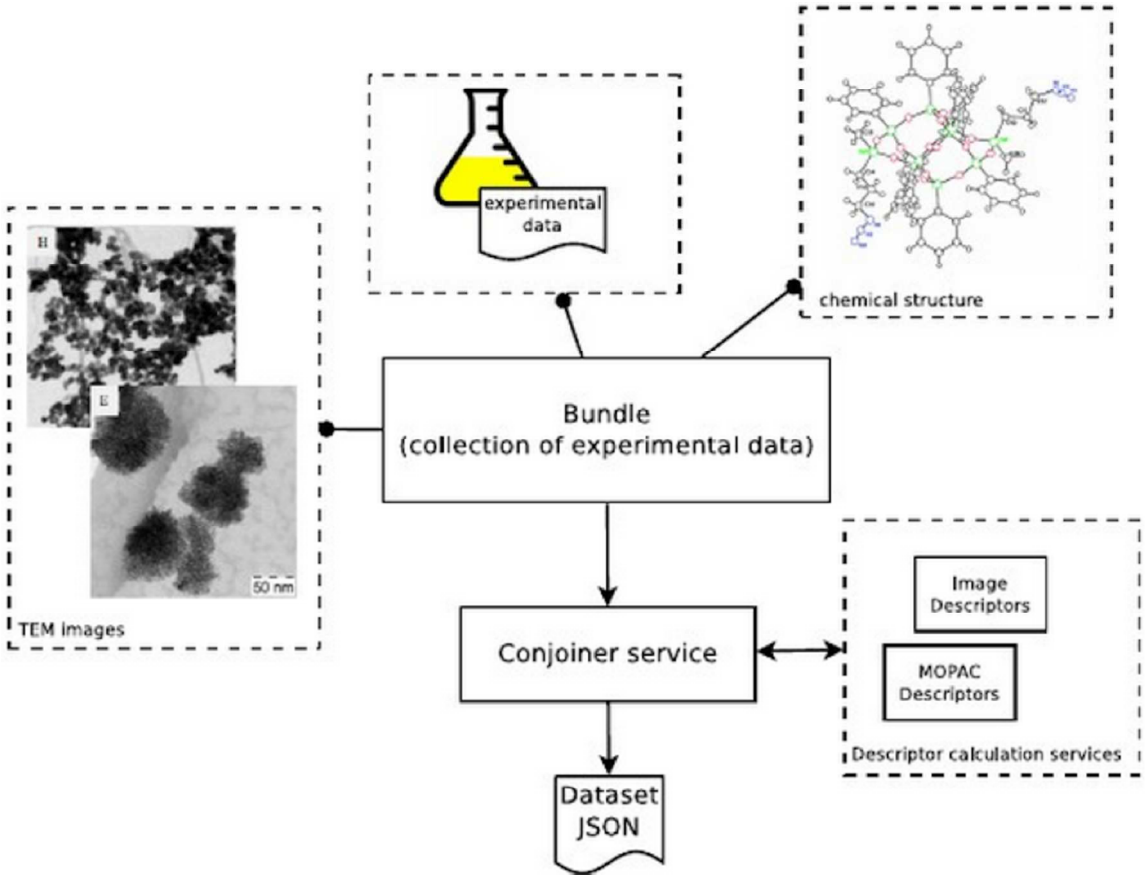

Figure 9. Schematic representation of Jaqpot Quattro's Conjoiner service. Heterogeneous data is processed and standardised for the algorithm services.

Algorithm integration for NanoQSAR Development. The Jaqpot Protocol of Data Interchange, in short JPDI, is a new feature of the JQ web services that allows developers of machine learning algorithms to integrate their implementations in the framework. This integration requires little engagement with intricate software development and allows algorithm developers to outsource their implementations and make them available to the nanomaterial design community through the eNanoMapper framework. The communication between eNanoMapper services and third-party JPDI services is carried out by exchanging JSON documents that contain no more information than what a modeling service needs to train a

predictive model, calculate descriptors, perform a prediction, evaluate the domain of applicability of a model or perform other tasks. Once a developer (possibly third-party) has prepared a JPDI-compliant web service, they need to register it with the eNanoMapper framework by specifying (i) the name of the algorithm, (ii) meta-data for the algorithm, such as a description, tags, copyright notice and any other meta-data supported by the Dublin core ontology (http://dublincore.org/) and/or the eNanoMapper ontology[27] (http://www.enanomapper.net/ontology), (iii) the URI of their implementation to be used as an endpoint for training, (iv) the corresponding URI for the prediction web service. The algorithm is then registered by POSTing a JSON file containing all this information to /algorithm. Once registered, the algorithm acquires a URI, and is exposed as a web service, that can be consumed by JQ. The JPDI protocol allows to dynamically and seamlessly incorporate any custom algorithmic implementation into eNanoMapper without any need for resource management (*i.e.*, the algorithm providers do not need to maintain a database system). The protocol specifies the form of data exchange between eNanoMapper services and third party algorithm web service implementations. The eNanoMapper framework already provides wrappers for WEKA[28] the R language[29] and the Python language.[30]

Integration with R. Integration with R is made possible through the OpenCPU (https://www.opencpu.org/) system, which defines a HTTP API for embedded scientific computing based on R, although this approach could be easily generalised to other computational back-ends.[31] OpenCPU acts as a wrapper to R that is readily able to expose R functions as RESTful HTTP resources. The OpenCPU server takes advantage of multi-processing in the Apache2 web server to handle concurrency. This implementation uses forks of the R process to serve concurrent requests immediately with little performance overhead. By doing so it enables access to those functions on simple HTTP calls converting R from a standalone application to a web service.

Integration with Python. The python web service implemented in this work was built using Flask (http://flask.pocoo.org/), a microframework based on the Jinja2 (http://jinja.pocoo.org/) and Werkzeug WSGI (http://werkzeug.pocoo.org/) libraries. Jinja2 is a template engine for python, whilst Werkzeug is a WSGI utility library, which incorporates very straightforward HTTP header parsing and easy-to-handle request/response objects. The entire web application fits nicely into a single python file. The application receives a JSON file containing a training request *via* a POST command containing a dataset, the feature for prediction and model parameters where applicable. The dataset field further contains the unique dataset URI and the data entry field filled with substance identifiers and property-value pairs. The application converts the fields into python objects or variables and calls the algorithm.

Integration with Weka. The Waikato Environment for Knowledge Analysis tool (Weka; http://www.cs.waikato.ac.nz/ml/weka/) is an open-source data mining software written in Java, featuring a vast collection of machine learning algorithms suitable for addressing a plethora of data mining problems. These include methods for classification, regression and clustering, as well as association rules among attributes. Weka also allows both supervised and unsupervised data pre-processing, ranging from data discretisation and mathematical operations, all the way to principal component projection and feature selection, using methods such as information gain. Finally, it includes a flexible data visualiser for both input and output.

Weka is highly popular due to its straightforward GUI, which allows all operations to be completed within a few clicks, as well as the number of widely used algorithms implemented and the variety of readable file formats allowed (such as ARFF and CSV). Each algorithm is initialised with suggested default values for its parameters, thus allowing the software to also be used by non-experts. However, for those who know the intricate details of each algorithm, the parameters can be adjusted for optimised results. It includes options for both internal cross-validation and a user-provided external validation test set, as well as statistical measures for assessing the derived model quality and predictive power. JQ provides a collection of basic algorithmic implementations under the JPDI protocol as a module of the core system, in order to provide a starting functionality as well as demonstrating the use of JPDI. This module gets shipped inside the Jaqpot Enterprise Application aRchive (EAR) bundle but does not utilise any of the Enterprise Java Bean (EJB) service layers of the core system, showing the exact same behaviour as expected of an external JPDI algorithm service, which boils down to being able to provide clean restful and stateless resources for each algorithm implementation that keep no internal state or data whatsoever. The module consists of a set of preprocessing algorithms and a domain of applicability calculation algorithm, coded by the Jaqpot development team, and a set of training algorithms, powered by the well-established machine learning libraries Weka (v 3.6.12) and LibSVM (v 3.17). Each algorithm has its own resource URI under the path */algorithms* and different functions for training and prediction mapped on */training* and */prediction* endpoints.

Conflict of interest

The authors declare no conflict of interest.

Acknowledgement

The eNanoMapper project has been funded by the European Union's Seventh Framework Programme for research, technological development and demonstration (FP7-NMP-2013-SMALL-7) under grant agreement no. 604134.

Author ORCID information

Charalampos Chomenidis (0000-0003-0855-9805)
Georgios Drakakis (0000-0002-6635-9273)
Georgia Tsiliki (0000-0001-7209-3670)
Evangelia Anagnostopoulou (0000-0002-9795-7452)
Angelos Valsamis (0000-0002-8921-2186)
Philip Doganis (0000-0002-0628-8434)
Pantelis Sopasakis (0000-0002-4584-2354)
Haralambos Sarimveis (0000-0002-8607-9965)

List of Abbreviations

Engineered nanomaterials (ENMs)
JaqPot Quattro (JQ)
REpresentation State Transfer (REST)
Application Programming Interfaces (APIs)

Predictive Model Markup Language (PMML)
Nano quantitative structure–activity relationship (nanoQSAR)
QSAR Prediction Reporting Format (QPRF)
Partial Least Squares (PLS)
Variable Importance in Projection scores (VIP) scores
Support Vector Machines (SVM)
Iterative Dichotomiser 3 (ID3)
Conditional Mutual Information decision trees (CMI)
Metal Oxide nanoparticles (MeOx)
Human keratinocyte cell line (HaCaT)
Principal Component Analysis (PCA)
Java Messaging System (JMS)
Java Non-blocking I/O (NIO)
Model-View-Controller (MVC)
HyperTextMarkup Language (HTML)
Cascading Style Sheets (CSS)
Asynchronous JavaScript and XML (AJAX)
Jaqpot Protocol of Data Interchange (JPDI)
Waikato Environment for Knowledge Analysis tool (Weka)
Enterprise Application aRchive (EAR)
Enterprise Java Bean (EJB)

References

1. Cao, Y.; Li, J.; Liu, F.; Li, X.; Jiang, Q.; Cheng, S.; Gu, Y. Consideration of Interaction between Nanoparticles and Food Components for the Safety Assessment of Nanoparticles Following Oral Exposure: A Review. *Environ. Toxicol. Pharmacol.* **2016**, *46*, 206–210.
2. Wolfram, J.; Zhu, M.; Yang, Y.; Shen, J.; Gentile, E.; Paolino, D.; Fresta, M.; Nie, G.; Chen, C.; Shen, H. et al. Safety of Nanoparticles in Medicine. *Curr. Drug Targets* **2015**, 16, 1671-1681.
3. Raj, S.; Jose, S.; Sumod, U. S.; Sabitha, M. Nanotechnology in Cosmetics: Opportunities and Challenges. *J. Pharm. BioAllied Sci.* **2012**, *4*, 186–193.
4. Fourches, D.; Pu, D.l Tassa, C.; Weissleder, R.; Shaw, S. Y.; Mumper, R. J.; Tropsha, A. Quantitative Nanostructure-Activity Relationship (QNAR) Modeling. *ACS Nano*, **2010**, *4*, 5703–5712
5. Puzyn, T.; Gajewicz, A.; Leszczynska, D.; Leszczynski, J. Nanomaterials -- the Next Great Challenge for Qsar Modelers. In *Recent Advances in QSAR Studies: Methods and Applications*; Puzyn, T., Leszczynski, J., Cronin, M. T., Eds.; Springer Netherlands: Dordrecht, **2010**, pp 383–409.
6. Gajewicz, A.; Rasulev, B.; Dinadayalane, T. C.; Urbaszek, P.; Puzyn, T.; Leszczynska, D.; Leszczynski, J. Advancing Risk Assessment of Engineered Nanomaterials: Application of Computational Approaches. *Adv. Drug Delivery Rev.* **2012**, *64*, 1663–1693.

7. Winkler, D. A.; Mombelli, E.; Pietroiusti, A.; Tran, L.; Worth, A.; Fadeel, B.; McCall, M. J. Applying Quantitative Structure-Activity Relationship Approaches to Nanotoxicology: Current Status and Future Potential. *Toxicology* **2013**, *313*, 15–23.
8. Winkler, D.A. Recent Advances, and Unresolved issues, in the Application of Computational Modeling to the Prediction of the Biological Effects of Nanomaterials, *Toxicol. Appl. Pharmacol.* **2016**, 299, 96-100.
9. Valsami-Jones, E.; Lynch, I.; Charitidis, C.A. Nanomaterial Ontologies for Nanosafety: A Rose by Any Other Name… *Nanomed Res. J.* **2016**, 3: 00070..
10. Sizochenko, N.; Leszczynski, J. Review of Current and Emerging Approaches for Quantitative Nanostructure-Activity Relationship Modeling: The Case of Inorganic Nanoparticles. *J. Nanotox. Nanomed.* **2016**, *1*, 1–16.
11. Jeliazkova, N.; Chomenidis, C.; Doganis, P.; Fadeel, B.; Grafström, R.; Hardy, B.; Hastings, J.; Hegi, M.; Jeliazkov, V.; Kochev, N. et al. The eNanoMapper Database for Nanomaterial Safety Information. *Beilstein J. Nanotechnol.* **2015**, *6*, 1609–1634.
12. *MOPAC*; Cambridge Soft Corporation: Cambridge, MA, **1996**.
13. Rasband W.S., ImageJ, U. S. National Institutes of Health, Bethesda, Maryland, USA (1997-2016).
14. Gajewicz, A.; Schaeublin, N.; Rasulev, B.; Hussain, S.; Leszczynska, D.; Puzyn, T.; Leszczynski, J. Towards Understanding Mechanisms Governing Cytotoxicity of Metal Oxides Nanoparticles: Hints from Nano-QSAR Studies. *Nanotoxicology* **2015**, *9*, 313–325.
15. Drakakis, G.; Moledina, S.; Chomenidis, C.; Doganis, P.; Sarimveis, H. Decision Trees for Continuous Data and Conditional Mutual Information as a Criterion for Splitting Instances. *Comb. Chem. High Throughput Screen.* **2016**, *19*, 423–428.
16. Puzyn, T.; Rasulev, B.; Gajewicz, A.; Hu, X.; Dasari, T. P.; Michalkova, A.; Hwang, H.-M.; Toropov, A.; Leszczynska, D.; Leszczynski, J. Using Nano-QSAR to Predict the Cytotoxicity of Metal Oxide Nanoparticles. *Nat. Nanotechnol.* **2011**, *6*, 175–178.
17. Walkey, C. D.; Olsen, J. B.; Song, F.; Liu, R.; Guo, H.; Olsen, D. W. H.; Cohen, Y.; Emili, A.; Chan, W. C. W. Protein Corona Fingerprinting Predicts the Cellular Interaction of Gold and Silver Nanoparticles. *ACS Nano* **2014**, *8*, 2439–2455.
18. Joint Research Centre QSAR Model Database and QSAR Model Reporting Formats, European Union Reference Laboratory for Alternatives to Animal Testing, Techical Report (2016). Available at: https://eurl-ecvam.jrc.ec.europa.eu/databases/jrc-qsar-model-database-and-qsar-model-reporting-formats (Date of access: 07/16/2017).
19. Lynch, I.; Weiss, C.; Valsami-Jones, E. A Strategy for Grouping of Nanomaterials Based on Key Physico-Chemical Descriptors as a Basis for Safer-by-Design {NMs}. *Nano Today* **2014**, *9*, 266–270.
20. Brandmaier, S.; Tetko, I. V. Robustness in Experimental Design: A Study on the Reliability of Selection Approaches. *Comput. Struct. Biotechnol. J.* **2013**, *7*, 1–10.
21. Brandmaier, S.; Sahlin, U.; Tetko, I. V.; Öberg, T. PLS-Optimal: A Stepwise D-Optimal Design Based on Latent Variables. *J. Chem. Inf. Model.* **2012**, *52*, 975–983.
22. Richardson L., Act Three: The Maturity Heuristic, 2008-QCon, San Francisco **2008**. Available at: https://www.crummy.com/writing/speaking/2008-QCon/act3.html (Date of access: 07/16/2017).
23. Holovaty, A.; Kaplan-Moss, J. *The Definitive Guide to Django: Web Development Done Right*; Apress: Berkeley, CA, **2009**.

24. David, F. *JavaScript: The Definitive Guide, 6th Edition*; O'Reilly Media, **2011**.
25. Chaffer, J.; Swedberg, K. *Learning jQuery - Fourth Edition*; Packt Publishing Ltd., **2013**.
26. Zakas, N. C.; McPeak, J.; Fawcett, J. *Professional Ajax, 2nd Edition*; Wrox (Wiley), **2007**.
27. Hastings, J.; Jeliazkova, N.; Owen, G.; Tsiliki, G.; Munteanu, C. R.; Steinbeck, C.; Willighagen, E. eNanoMapper: Harnessing Ontologies to Enable Data Integration for Nanomaterial Risk Assessment. *J. Biomed. Semant.* **2015**, *6*, 10.
28. Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I. H. The WEKA Data Mining Software: An Update. *SIGKDD Explor.* **2009**, *11*, 10–18.
29. R. Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing (Vienna, Austria. 2015). Available at: http. www.R-project.org (Date of access: 07/16/2017).
30. Python Software Foundation. Python Language Reference Manual (version 2.7). Available at: http://www.python.org (Date of access: 07/16/2017).
31. Ooms, J. The OpenCPU System: Towards a Universal Interface for Scientific Computing through Separation of Concerns, *Preprint arXiv*:1406.4806. **2014**, No. 2000, 1-23.

GRAPHICAL ABSTRACT.